CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

cspit
Charusat Shri C. J. Patel Institute of Technology

CE144
OBJECT ORIENTED PROGRAMMING WITH C++

UNIT-1
Principles of object-oriented Programming

N. A. Shaikh
nishatshaikh.it@charusat.ac.in

Topics to be covered

- **Basic concept of object-oriented Programming**
- **Benefits of OOP**
- **Difference between object oriented language and procedure oriented language**

Computer Programming

- An **algorithm** is a step-by-step process
- A **computer program** is a step-by-step set of instructions for a computer

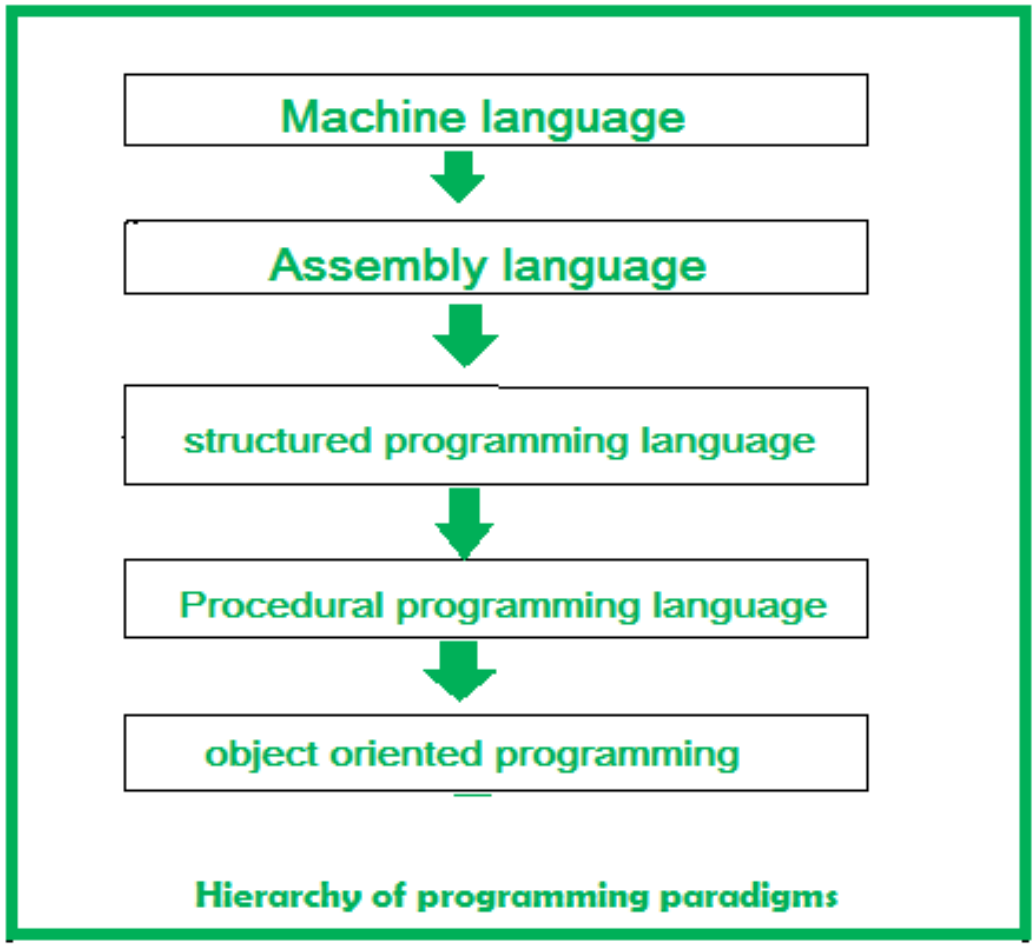
Programming Languages:

- Machine languages (Ex: 1110100010101)
- Assembly languages (Ex: ADD 1001010 , 1011010)
- High-Level languages (grosspay=basepay+overtimepay)

High-Level languages are mainly divided into,

1. Procedure oriented language(C)
2. **Object oriented language(C++)**

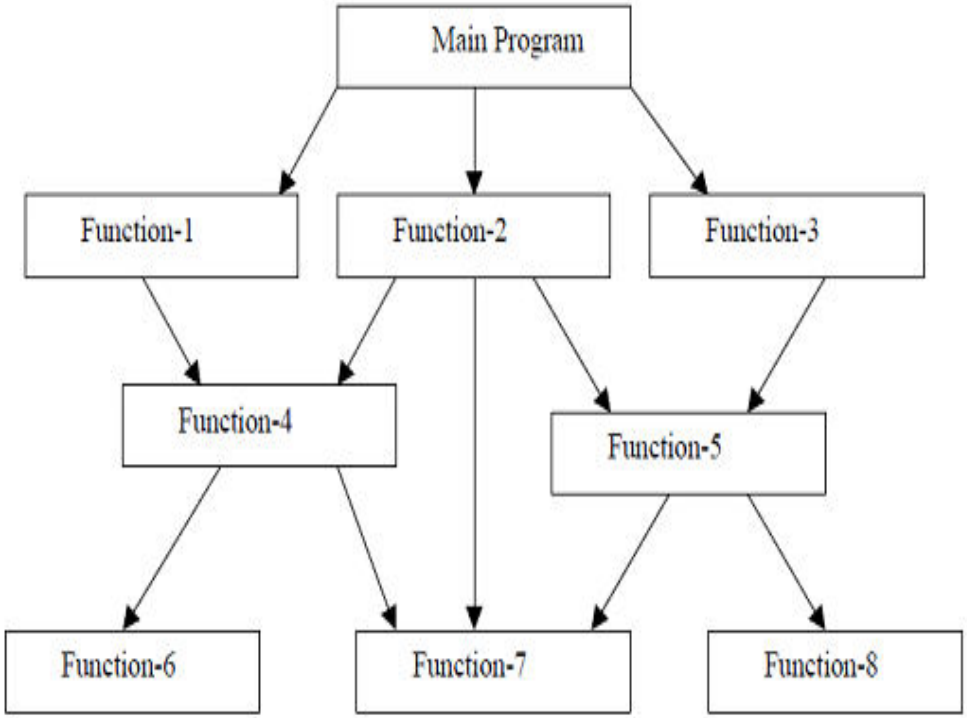
Computer Programming



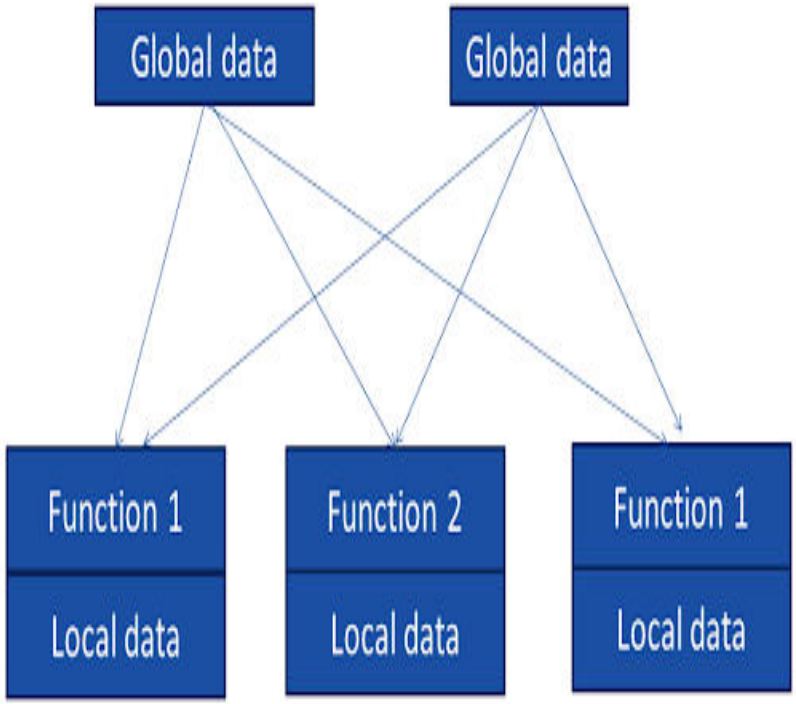
Hierarchy of programming paradigms

Procedure oriented Programming

- Conventional programming, using high level languages such as COBOL, FORTRAN and C, is commonly known as **procedure oriented programming (POP)**.



Structure of procedural oriented programs



Procedure oriented Programming

Characteristics:

- **Emphasis** is on doing things.
- Large programs are divided into smaller programs known as **functions**.
- Primary focus is on **functions** rather than data.
- Most of the functions share **global** data.
- Data move openly around the system from function to function
- Functions transform data from one form to another.
- Employs **top-down approach** in program design.

Need of Object-oriented Programming

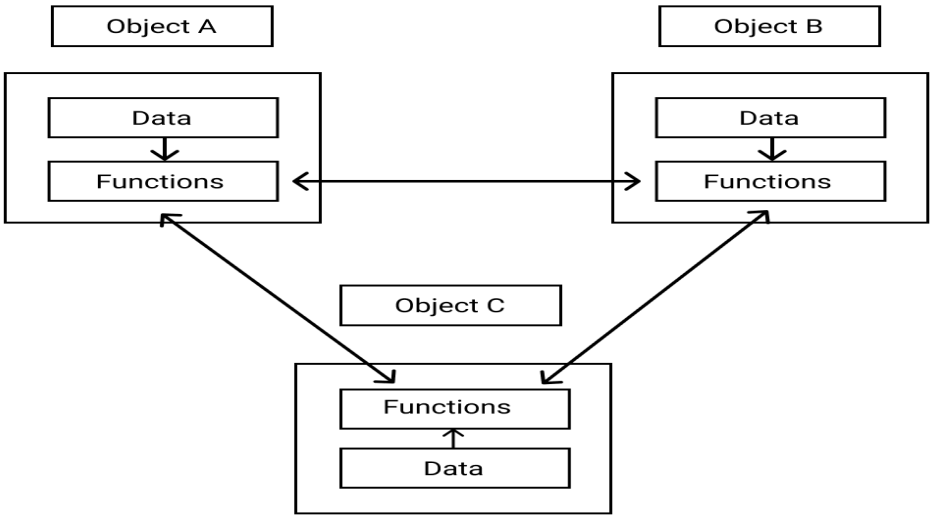
- Object-oriented programming was developed because limitations were discovered in earlier approaches to programming.

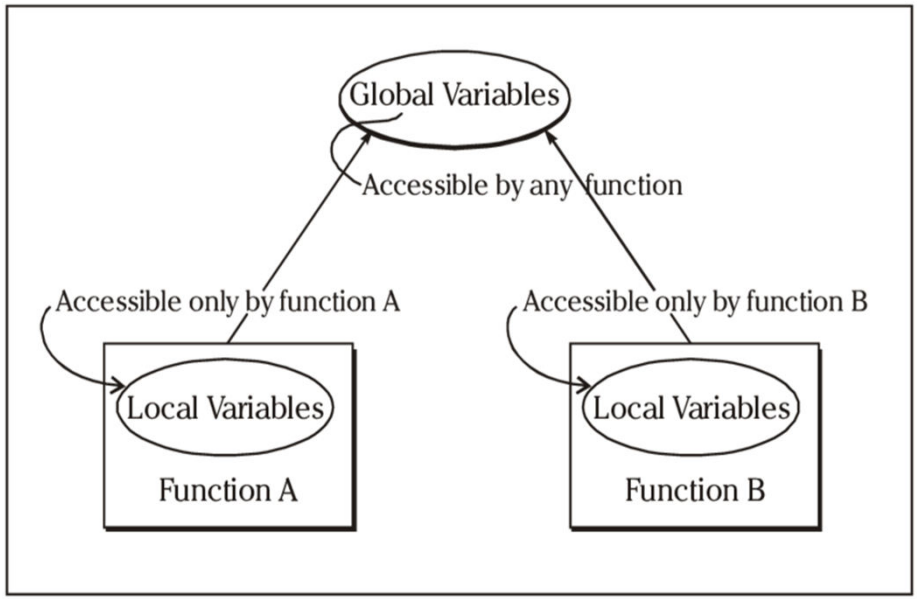
Limitation of Procedure-oriented Programming:

- Procedural codes are very difficult to maintain, if the code **grows**
- Data is exposed to whole program, so **no security** for data.
- Difficult to relate with **real world objects**.
- Difficult to create **new data types**.
- Importance is given to **operations** on data rather than data.

Object-oriented Programming

- Object-oriented programming is a programming paradigm based on the concept of "**objects**", which can contain **data** and **code**: data in the form of fields, and code, in the form of procedures.
- A feature of objects is that an object's own procedures can access and often modify the data fields of itself



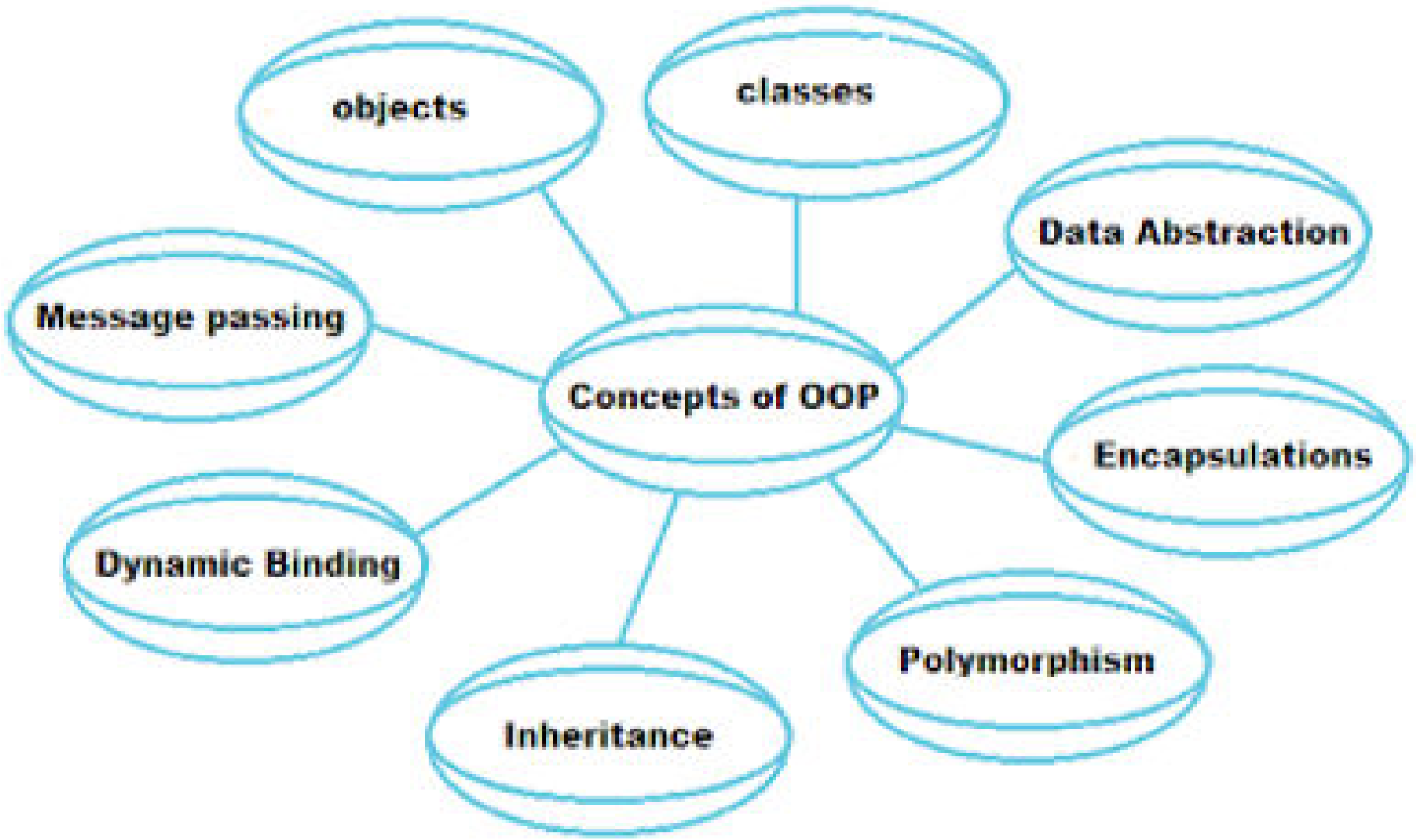


Object-oriented Programming

Characteristics:

- Emphasis on **data** rather than procedure
- Programs are divided into entities known as **objects.**
- Does not allow data to flow freely around the system
- Functions that operate on data of an object are tied together in data structures.
- Data is **hidden** and cannot be accessed by external functions.
- Objects communicate with each other through functions.
- New data and functions can be easily added whenever necessary.
- Follows **bottom up design** in program design.

Basic Concept of Object-oriented Programming



Objects

- Any entity that has state/characteristics and behavior is known as an **object**.
- **For example:** chair, pen, table, keyboard, bike etc.
- It can be physical and logical.
- **An Object is an instance of a Class**.
- When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) **memory** is allocated.
- When a program is executed the objects interact by sending messages to one another.

Objects(Cont..)

- Each object contains data and code to manipulate the data.
- Objects can interact without having to know details of each other's data or code.
- It is sufficient to know the type of message accepted and type of response returned by the objects.
- **There can be many objects of same class.**

Classes

- The building block of C++ that leads to Object-Oriented programming is a **Class**.
- A Class is a user-defined data-type and behave like the built-in types of a programming language which has data members and member functions.
- **A class is like a blueprint for an object.**
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions define the properties and behavior of the objects in a Class

Classes(Cont..)

- **For Example:** Consider the Class of Cars. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. So here, Car is the class and wheels, speed limits, mileage are their properties.
- Class is a blue-print representing a group of objects which shares some common properties and behaviors.
- **For examples,** Mango, Apple and orange members of class fruit.
- The syntax used to create an object

Fruit Mango;

Data Abstraction

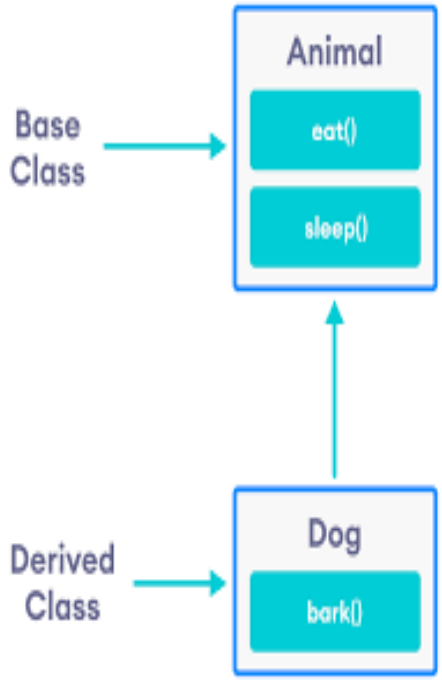
- **Abstraction** refers to the act of representing essential features without including the background details or explanation/implementation.
- Consider a real life **example** of a man driving a car The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car This is what abstraction is.
- Since the classes uses the concept of data abstraction, they are known as **Abstract Data Types (ADT)**.

Encapsulation

- The wrapping up of data and function into a single unit (called class) is known as **encapsulation**.
- The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.
- Encapsulation leads to **data hiding**(insulation of the data from direct access by the program)
- In OOP we achieve encapsulation by making data members as private and having public functions to access these data members.

Inheritance

- **Inheritance** is the process by which objects of one class acquired the properties of objects of another classes.
- The class that inherits properties from another class is called **Sub class/Derived Class/ child class.**
- The class whose properties are inherited by sub class is called **Base Class/Super class/ parent class**
- The principal behind this sort of division is that each derived class shares common characteristics with the class from which it is derived

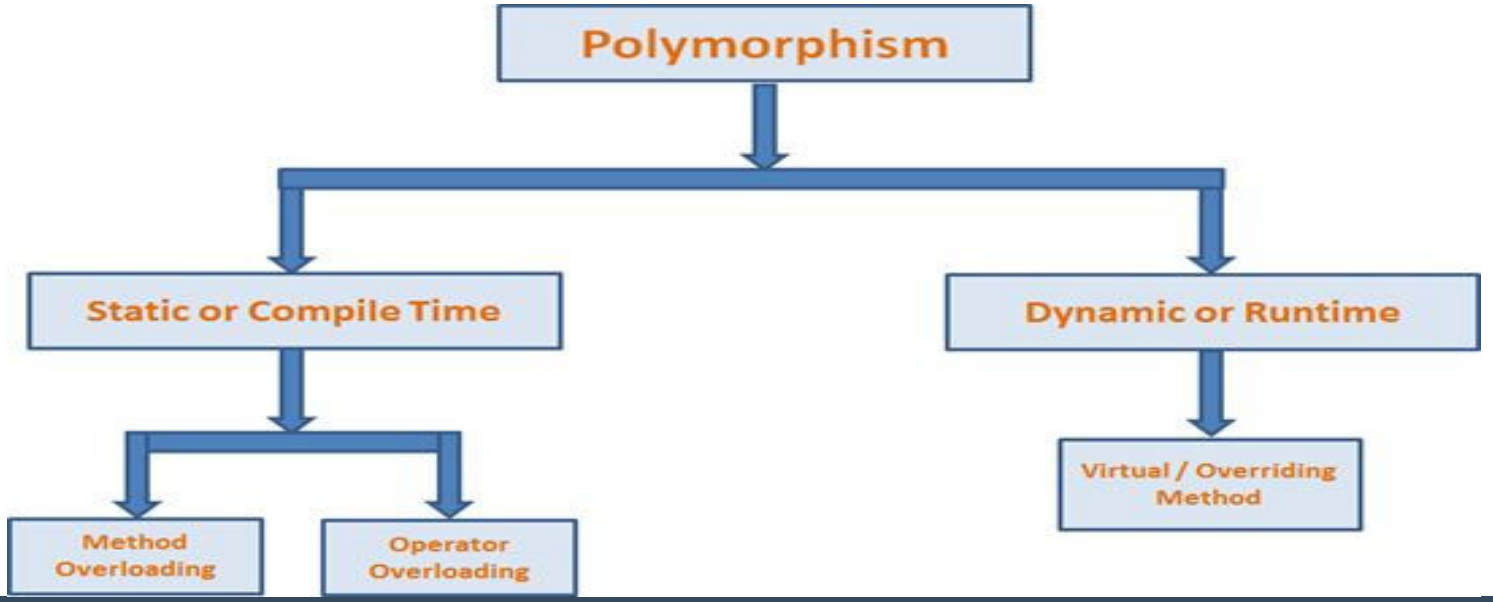


Inheritance(Cont..)

- In OOP, the concept of inheritance provides the idea of **reusability**.
- This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined feature of both the classes.
- Inheritance is basically used for reducing the overall code size of the program.

Polymorphism

- **Polymorphism**, a Greek term, means the ability to take more than one form.
- **For example,** consider the operation of addition. For two numbers, the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation.



Static Binding

- **Binding** refers to the linking of a procedure call to the code to be executed in response to the call.
- **Early Binding (compile-time time polymorphism):** As the name indicates, compiler (or linker) directly associate an address to the function call.
- It replaces the call with a machine language instruction that tells the mainframe to leap to the address of the function.
- By default early binding happens in C++. Late binding is achieved with the help of **virtual keyword**

Dynamic Binding

- **Dynamic binding** means that the code associated with a given procedure call is not known until the time of the call at run time
- **Late Binding (Run time polymorphism):** In this, the compiler adds code that identifies the kind of object at runtime then matches the call with the right function definition. This can be achieved by declaring a virtual function.

Message Passing

- In OOP, objects communicate with each other using messages.
- When objects communicate, information is passed back and forth between the objects.
- A message for an object is a **request** for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results.
- A message generally consists of the **object name, method name and actual data** that is to be sent to another object.
- Object has a life cycle. They can be created and destroyed.
- Communication with an object is feasible as long as it is alive.

Benefits of OOP

- Through **inheritance**, we can eliminate **redundant code** and extend the use of existing classes.
- We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development **time and higher productivity**.
- The principle of **data hiding** helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- It is possible to have multiple instances of an object to co-exist without any interference.

Benefits of OOP(Cont..)

- It is possible to map objects in the problem domain to those in the program.
- It is easy to **partition the work** in a project based on **objects**.
- The data-centered design approach enables us to capture more details of a model in implementable form.
- Object-oriented systems can be easily upgraded from **small to large systems**.
- **Message passing** techniques for communication between objects makes the interface descriptions with external systems much simpler.
- **Software complexity** can be easily managed.

POP VS OOP

	Procedure Oriented Programming	Object Oriented Programming
Divided Into	In POP, program is divided into small parts called **functions**.	In OOP, program is divided into parts called **objects**.
Importance	In POP, Importance is not given to **data** but to functions as well as **sequence** of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a **real world**.
Approach	POP follows **Top Down approach.**	OOP follows **Bottom Up approach**.
Access Specifiers	POP does not have any access specifier.	OOP has access specifiers named **Public, Private, Protected**, etc.
Data Moving	In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.

POP VS OOP

	Procedure Oriented Programming	Object Oriented Programming
Expansion	To add new data and function in POP is **not so easy**.	OOP provides an easy way to add new data and function.
Data Access	In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOP, data can not move easily from function to function,it can be kept public or private so we can control the access of data.
Data Hiding	POP does not have any proper way for hiding data so it is **less secure**.	OOP provides Data Hiding so provides **more security**.
Overloading	In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.
Examples	Example of POP are : **C, VB, FORTRAN, Pascal.**	Example of OOP are : **C++, JAVA, VB.NET, C#.NET.**

Object-based VS Object-Oriented Programming Languages

Object Based Languages

- Object based languages supports the usage of object and encapsulation.
- They does not support inheritance or, polymorphism or, both.
- They supports built-in objects.
- JavaScript, VB are the examples of object based languages.

Object Oriented Languages

- Object Oriented Languages supports all the features of Oops including
- inheritance and polymorphism.
- They does not supports built-in objects
- C#, Java, VB. Net are the examples of object oriented languages.

Object-based VS Object-Oriented Programming Languages

Object-Oriented = Object based features

+

inheritance

+

polymorphism

Application of OOP

- Real-time systems
- Simulation and modeling
- Object-oriented databases
- Hypertext, hypermedia and expertext
- Al and expert systems
- Neural networks and parallel programming
- Decision support and office automation systems
- CIM/CAM/CAD systems

End of Unit-1